Final Project: Classification and Detection with Convolutional Neural Networks

HAN UL (BRIAN) LEE
hanbrianlee@gatech.edu

1 INTRODUCTION

I have tried various approaches to tackle this project. For classification, I have tried implementing an approach similar to GoodFellow's paper (Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2014) for detecting multi-digit sequence directly, as well as detecting individual digits separately and then combining the digits. For individual digits classifier, I have used off the shelf architecture VGG16 (Simonyan & Zisserman, 2014) and tried training it from scratch as well as from ImageNet (Deng et al., 2009) pretrained weights. For detection, I have tried implementing sliding-windows with binary digits/no digits classifier (for multi-digit sequences to be used in conjunction with GoodFellow's approach), sliding-windows binary digit/no digit classifier (for individual digit classifiers), and MSER (Maximally Stable Extremal Regions) (Matas, Chum, Urban, & Pajdla, 2002) approach for blob detections. In the training of all networks described in this report, data augmentations of 20 degrees random rotations and random resized crops between scale 0.9 1.0 and 0.75 1.25 ratios were done, with normalization.

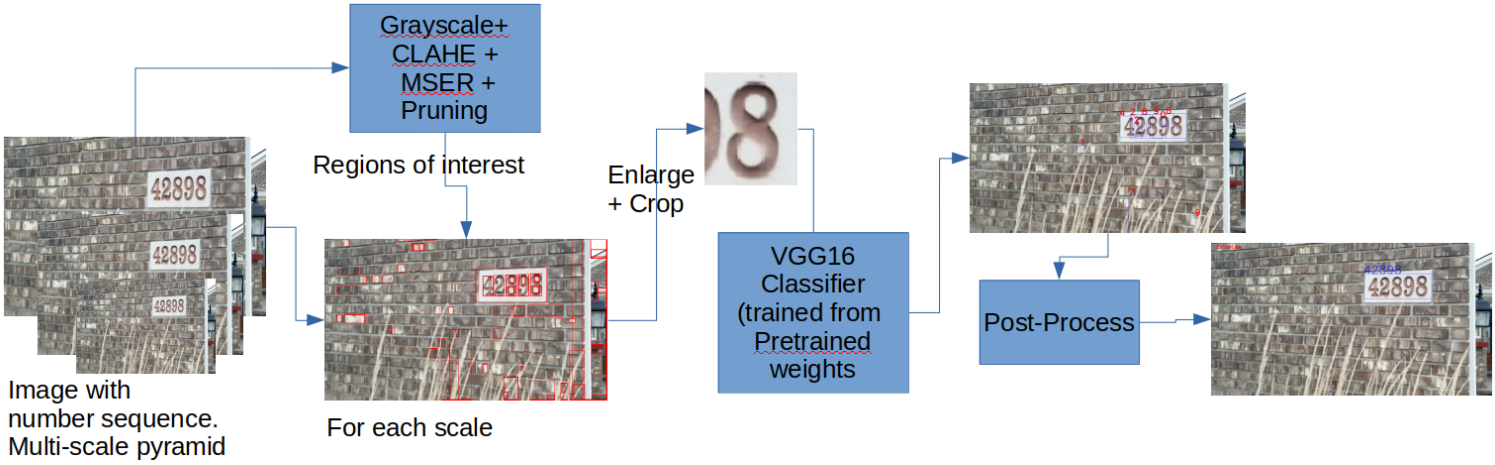

Figure 1: Pipeline

2 DATASET

SVHN Dataset (Netzer et al., 2011) was utilized for this project which contains house numbers in various scenes and provides format1 with the scene with multiple boxes in them, and format2 which already have cropped digits. Format1 was utilized for negative example generations, binary neural network classifiers for region of interest extraction, and GoodFellow's approach data preparation. Format2 was utilized for single-digit classification neural network training.

3 METHODS AND IMPLEMENTATION

3.1 Classification

3.1.1 *Approach similar to GoodFellow's Multi-digit Number Recognition*

This approach, while still requiring a good region detector to propose the center of the digit sequences, removes the need of detecting individual digits and then performing clustering, non-max suppression, and combining them. It directly gets as input the sequence of numbers, and after extracting feature maps through convolution layers, go through separate fully connected layers to reason how long the sequence is, as well as each individual number's classes in the sequence. By training this classifier against the sum of log losses from each of these fully

connected layers (which is equivalent to multiplied softmax probabilities of each layers, and minimizing negative log likelihoods by summing up cross entropies is the same as maximizing the log likelihoods described in the paper), the classifier's feature map extraction layers learn all necessary features to aid each fully connected layers correctly classify their individual goals. In other words, the feature extraction layers will learn all necessary edge and shape filters necessary to separate each digit and unique enough representations for each type of digit so that the digit classification fully connected layers to make their decision. The novelty of this network is that by training on sequence of digits, the fully connected layers for each digit in the sequence will each learn to connect relevant feature maps automatically (i.e. the fully connected layers for the first digit in the sequence of 5 will learn the weight values to link the whole bounded region's extracted features to classify the first digit location's digit class. This means that the weights connecting feature pixel locations on the far right will most likely be low and high for the left-most locations, for the first digit location).
As for the detailed network architecture, I did not exactly follow the architecture described in the paper, but rather a shallower version of it. I would have implemented a more deeper version as described in the paper if I didn't move onto another approach (Figure 1) for the reasons described at the end of this paragraph, so I stopped. The training for this network was done with stochastic mini-batch gradient descent, and the ground-truth data were collected via a script from SVHN dataset format1. For positive examples, simply the ground-truth bounding boxes locations were merged to form the bigger bounding box (as the dataset has only one sequence per image), and for negative examples, random cropped regions with no intersection with ground-truth boxes were collected, and the total datasets were balanced between positives and negatives to be of the same size. Balancing between sequence lengths or digit classes contained inside were not done as it is not trivial and also I moved onto a different approach after trying this approach on house numbers around me (which have 5 digit sequences). The network mostly successfully detected up to 4 digits on test images as the training set did not have many 5 digit sequence examples. See Figure 4 for more details.

3.1.2 *VGG16 Architecture Approach for single-digits* - **final approach**

VGG16 is one variant of VGG architectures invented by Visual Geometry Group from Oxford University. It has many variants such as VGG11, VGG19, etc. For this project, VGG16 was prescribed and I trained the architecture both from scratch (random weight initialization) and from transfer-learning from ImageNet pretrained weights available via torchvision. VGG16 has been one of the most popular backbones, probably second to ResNet architectures. Its deep layers compared to Alexnet (Krizhevsky, Sutskever, & Hinton, 2012) or LeNet (LeCun, Bottou, Bengio, & Haffner, 2001) enable the network to capture deeper features necessary for various tasks, but contain issues mostly with vanishing gradients due to such deep architecture. Batch-norm layers can help with convergence by having each layers' values to be normalized to one standard deviation so that every node has better chances of updating its weights by the backpropagation with less chances of exploding or vanishing gradients, and batch-norm variants of VGG exist on torchvision (not used for this project). Also, lacking the skip connections (such as those in ResNet (He, Zhang, Ren, & Sun, 2015)) makes the deeper layers to not really make connections with earlier layers, preventing the network to learn more coherent features across layers. Even though VGG16 has its own deficiencies, it has been proven to work for a lot of tasks and for the rather simple problem of detecting only digits, I deemed it sufficient and the training was very quick as it only had 11 classes to train. I added one negative class to the existing 10 classes that are gathered by random crops from SVHN dataset format1, complying to the following conditions:

- IoU with any ground-truth single-digit box is not greater than 0.3
- Ground-truth single-digit box is not encompassed by the random crops.

The training data were balanced during the training phase, by first computing the distributions and assigning probabilities for each class being picked, the minibatch data fed for training was always balanced amongst the 11 classes. I did it this way instead of balancing out the data before feeding to dataloaders to maximize the variation in the data shown to the network. Training was again done by mini-batch stochastic gradient descent method. The training done with transfer learning from ImageNet prevailed over the one trained from scratch by approximately 1% higher accuracy (see Table 1) in test-set thanks to being already pre-trained in much larger dataset with more more diversity in the ImageNet dataset, and was therefore selected as my final classifier.

3.2 Detection (i.e. Localization)

3.2.1 *Sliding-windows sequence of digits existence classifier*

This detector was designed to go with the Goodfellow's approach. I used the same feature extraction part as Section 3.1.1's, but for the fully connected layer, a binary classification task was performed determining whether a sequence of numbers is enclosed by the sliding window or not (see Section 3.2.1. Positives and negatives were collected via random crops from SVHN dataset format1 again. Random crops with more than 0.7 IoU - intersection over union - (following Faster-RCNN's paper (Ren, He, Girshick, & Sun, 2015)) were assigned positive labels, while the following conditions were used for negative labels.

- IoU with the ground-truth sequence box is not greater than 0.3
- Ground-truth sequence box is not encompassed

Random crops that do not meet either of the positive label or negative label conditions were disregarded, as was done in the Faster-RCNN anchor based RPN (Region Proposal Network) detector (Ren et al., 2015). The pipeline with this detector was pretty simple as the windows with higher than certain softmax probability for the "exists" class could be passed to the classifier for multi-digit sequence classification. Amongst multiple positive detections, the assumption of only one digit-sequence in the image was used and the detection with the highest classification score (multiplication of digit sequence logit times the respective digits' logit values) was selected to be drawn the final bounding box around. Sliding windows were formed into batches for faster processing.

3.2.2 *Sliding-windows single digit existence classifier*

A simple CNN binary classifier was designed using some convolution layers, batch-norm layers, drop-out and fully connected layers. The job was to simply detect whether a number exists inside the sliding window or not. The ratio of the sliding window was selected to reflect most digits (about 1.5:1 height:width ratio), and was resized to 32x32 to be fed into this classifier. The sliding windows were batched and passed to the classifier mentioned in Section 3.1.2 for classification. The positive and negative random samples were selected the same way described in Section 3.2.1. This approach was quite robust but speed was slow. Since more efficient structures who share feature maps across different anchors and scales such as SSD (Liu et al., 2016), YOLO (Redmon, Divvala, Girshick, & Farhadi, 2015), Faster-RCNN (Ren et al., 2015) were banned for this project, the detections had to rely more on explicit multi-scale pyramids and sliding window strides and sizes. The need of more fine-grained sliding window sizes, pyramids, and strides, caused slowness and pushed me to try another approach mentioned in Section 3.2.3. After detection of single digits are done, post-processing is done to weed out outliers and combine the remainders, which are described in Section 3.3.

3.2.3 *MSER (Maximally Stable Extremal Regions)* - **final approach**

This approach really surprised me by its efficacy and thus got chosen as my final approach. MSER utilizes multi-level thresholds (intervals between thresholds controlled by delta parameter) and by drawing boundaries based on connected components to select regions that survive certain numbers of thresholds. Its strength is in affine invariance and to some extent size invariant (which was solved via multi-scale image pyramids) to select regions between minimum and maximum areas. Its processing speed was very quick but it did return many spurious detections with various sizes especially in areas with trees or other clutters in the background, which were later pruned by the classifier and post-processing step (see Section 3.3). Since MSER relies mostly on pixel intensity thresholding,
I improved the contrast before feeding into MSER by CLAHE (contrast limited adaptive histogram equalization) and gray-scale images were used. Also, To make the detection more robust, I had to lower delta values to capture more detections (even though it invites more false-positives), and at the cost of pipeline speed, I increased robustness of MSER detection. To combat false positives, I've pruned away those smaller than 2 3 pixels (as doing any classification on 2x2 or 3x3 pixels won't be trust-worthy anyway), and checked if the height to width ratio is properly suited for digit-type shapes (considering certain degree of worst-case rotations). I also had to increase the bounding box in all dimensions relative to the width and height ratios of MSER proposals, as the training dataset did not always have very tight bounding boxes, but rather loosely bounding boxes. This

Figure 2: Positive Results. Left: Successfully detect even small scales and lots of number looking like clutters in the background. Right: Even with big rotations and lots of brick clutter in the background, correctly classifies, filters, and combines numbers.

Figure 3: Negative Results. Left: Missed 4 due to skewdness of sequence causing clustering algorithm to detect 4 as an outlier. Right: MSER detection failed to propose any of the numbers due to lack of contrast between numbers and the immediate background.

especially helped for classifying digit class ”1”s as a tight MSER box around 1, when stretched to 32x32, looks mostly a filled-up blob, and not much shape information can be extracted. For fine-tuning, I used CV trackbar to tune against multiple various images for min-area, max-area, max-variation, delta values along with CLAHE parameters, by visualizing immediate changes to MSER proposals.

3.3 Post-processing

I relied on the assumption that the number sequences are horizontally arranged regardless of their orientation. After MSER + VGG classifier, multiple detections including false positives arrive. First, the detections with class 10 (no number) are pruned and then are thresholded to 0.99 (my VGG classifier had strong accuracy). Then Non-Max Suppression with 0.4 IoU was done and also a secondary custom intersection-ratio only non-max suppression was done to take care of situations where a number exists inside another big one whose IoU wouldn’t be high but one of them does need to be pruned. And then Density-based spatial clustering of applications with noise (DBSCAN) was used to cluster numbers with the distance criteria defined by the average of all detection box widths. The longest cluster with the lowest total distances was selected (as number sequence is more than 1 digit, and the longer the more likely the correct sequence). Afterwards, with the selected cluster, another DBSCAN clustering followed using euclidean distance value computed from at most 5 detections (after being outlier-pruned based on mean and variance), to acquire the final cluster. The final cluster of detections was ordered based on horizontal axis and then grouped together to form the final sequence.

4 RESULTS AND ANALYSIS

Positive results are shown in Figure 2, which showcases small numbers in the scene and severely rotated sequences being detected properly. Negative results shown in Figure 3 exploit the weakness of the my pipeline approach; the left one, while successfully detecting all numbers (42886), also detected smaller detections (0’s which were parts of 8’s) and the post-processing clustering algorithm has failed to weed out the 0’s due to

skewdness of the numbers causing widths-based clustering to miss the ”4” in the beginning. With little bit of fine-tuning, the correct output could be yielded but it could break otherwise positive examples. On the right is a case where MSER has failed to propose any of the numbers of candidates, due to not clear enough contrast. This again can be solved by tuning CLAHE parameters, but doing so may introduce more false positives on otherwise good examples.
On another note, I believe utilizing my simple binary classifier from Section 3.2.2 would have correctly proposed the numbers despite the contrast, with fine-grained multi-scale pyramids and window sizes. However, the speed would have been very slow to keep that approach robust and generalize, as it would require many more fine-grained scales, window-sizes and small stride. Tuning delta to be lower would also have caught the numbers, but it would in return more false positives and make the whole pipeline slower yet again. So in the end, the performance difference may have been similar, but since MSER returns tighter boxes without use of multi-scale window sizes, I would still think MSER would be faster. State of the arts such as Faster-RCNN (Ren et al., 2015) combined with Feature Pyramid Networks scheme (Lin et al., 2017) will be more robust and faster than my pipeline. These architectures utilize information not only bounded by the window sizes and also are able to effectively mix local and global semantic information to correct for tighter bounding boxes by regressing corrective box coordinates and sizes, on top of classifying foreground vs background, as well as classifying object class. Also feature pyramids will help detect smaller numbers, and mixing image pyramids as well, accuracy will be even higher.
Within the confines of this project’s scope, my pipeline can be improved by more fine-tuning, but there will be a limit to its performance as hand-crafted feature tuning has limitations. On the flip side, neural network-learned features are limited by lack of well labeled, well distributed datasets. So perhaps for limiting dataset to SVHN dataset (Netzer et al., 2011), more fine-tuning parameters with the current pipeline would be one of the best approaches.

5 PERFORMANCE STATISTICS

Cross-validation and loss curves for my network, VGG trained from scratch, and VGG trained from ImageNet (Deng et al., 2009) pretrained weights, are shown in Figure 4, Figure 5, Figure 6, Figure 3.2.2, and Figure 3.2.1 along with their test scores in Table 1. Epochs at which weights are frozen are chosen based on cross-validation loss curves and a little bit based on accuracy curve, before train-set and validation-set results deviate too far away from each other, to avoid over-fitting.

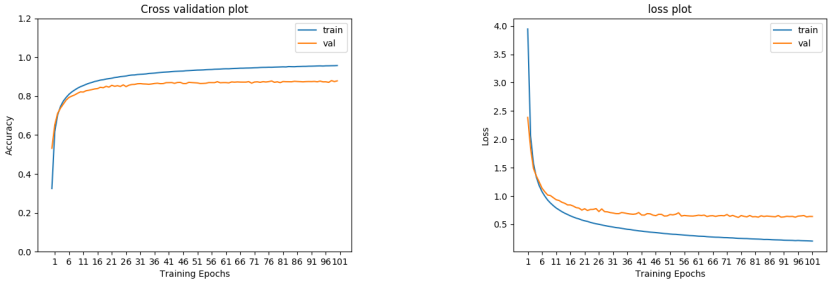

Figure 4: My own network. Learning-rate: 0.0005, Batch-size: 128

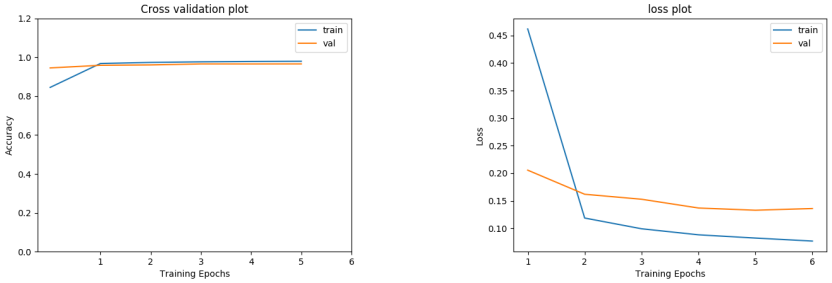

Figure 5: VGG trained from scratch. Learning-rate: 0.005. Batch$_size$: 256

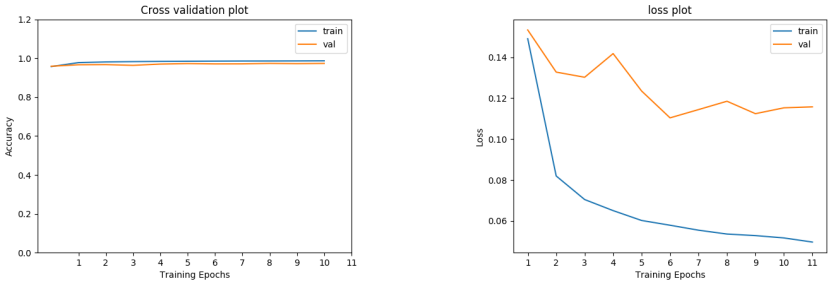

Figure 6: VGG trained from pre-trained weights. Learning-rate: 0.005. Batch$_size$: 256

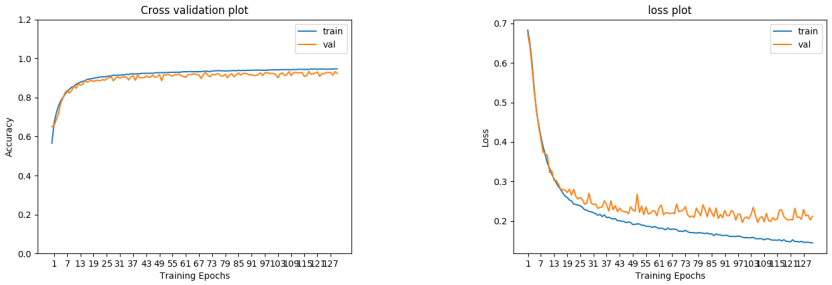

Figure 7: Single-digit detector. Learning-rate: 0.0004. Batch$_size$: 256

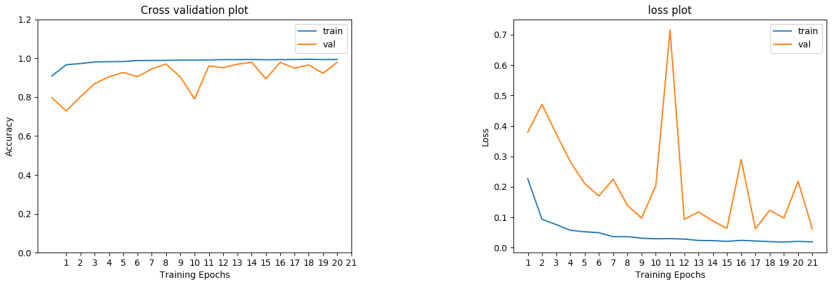

Figure 8: Multi-digit detector. Learning-rate: 0.001. Batch$_size$: 256

Table 1: Neural Network Test Performances against their respective test sets (50% of SVHN format 1 dataset's test split). MSER is not a network so it does not have a metric against test sets. It was visually verified against test images and videos instead.

Classifier	Chosen Epoch	Test Cross-Entropy Loss	Test Accuracy %
Multi-digit GoodFellow	50	0.6941	87.33
VGG16 from scratch	4	0.1271	96.85
VGG16 from pre-trained	4	0.1074	97.47
Detector	**Chosen Epoch**	**Test Cross-Entropy Loss**	**Test Accuracy %**
Multi-digit detector	20	0.0631	97.90
Single-digit detector	100	0.1849	93.27

REFERENCES

1. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Cvpr09.*
2. Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Iclr2014.*
3. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. Retrieved from http://arxiv.org/abs/1512.03385
4. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
5. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (2001). Gradient-based learning applied to document recognition. In *Intelligent signal processing* (p. 306-351). IEEE Press.
6. Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., & Belongie, S. J. (2017). Feature pyramid networks for object detection. In *Cvpr* (p. 936-944). IEEE Computer Society. Retrieved from http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#LinDGHHB17
7. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector.. Retrieved from http://arxiv.org/abs/1512.02325 (To appear.)
8. Matas, J., Chum, O., Urban, M., & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In P. L. Rosin & A. D. Marshall (Eds.), *Bmvc.* British Machine Vision Association. Retrieved from http://dblp.uni-trier.de/db/conf/bmvc/bmvc2002.html#MatasCUP02
9. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Nips workshop on deep learning and unsupervised feature learning 2011.* Retrieved from http://ufldl.stanford.edu/housenumbers
10. Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, *abs/1506.02640*. Retrieved from http://arxiv.org/abs/1506.02640
11. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks..
12. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*. Retrieved from http://arxiv.org/abs/1409.1556